Intel® PXA27x Processor Family

Specification Update

October 2004

.**Notice**: The Intel® PXA27x Processor Family may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Order Number: 280071-003

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

MPEG is an international standard for video compression/decompression promoted by ISO. Implementations of MPEG CODECs, or MPEG enabled platforms may require licenses from various entities, including Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at http://www.intel.com.

AlertVIEW, i960, AnyPoint, AppChoice, BoardWatch, BunnyPeople, CablePort, Celeron, Chips, Commerce Cart, CT Connect, CT Media, Dialogic, DM3, EtherExpress, ETOX, FlashFile, GatherRound, i386, i486, iCat, iCOMP, Insight960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel ChatPad, Intel Create&Share, Intel Dot.Station, Intel GigaBlade, Intel InBusiness, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetStructure, Intel Play, Intel Play logo, Intel Pocket Concert, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel TeamStation, Intel WebOutfitter, Intel Xeon, Intel XScale, Itanium, JobAnalyst, LANDesk, LanRover, MCS, MMX, MMX logo, NetPort, NetportExpress, Optimizer logo, OverDrive, Paragon, PC Dads, PC Parents, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, ProShare, RemoteExpress, Screamline, Shiva, SmartDie, Solutions960, Sound Mark, StorageExpress, The Computer Inside, The Journey Inside, This Way In, TokenExpress, Trillium, Vivonic, and VTune are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Copyright © Intel Corporation, 2004. All Rights Reserved.

*Third-party brands and names are the property of their respective owners.

Revision History

Revision Date	Version	Description
October, 2004	003	Added Errata 51, 52, 53, 54
August, 2004	002	Added Errata 48, 49, 50 Added Documentation Changes 5 - 15
April, 2004	001	First publication

Preface

This document contains updates to the specifications for the Intel® PXA27x Processor Family, listed in Table 1. This document is a compilation of device and documentation errata, specification clarifications, and specification changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, and tools.

Intel Corporation has endeavored to include all documented errata in the consolidation process. However, Intel makes no representations or warranties concerning the completeness of the ***Intel® PXA27x Processor Family Specification Update***.

Information types defined in Nomenclature are consolidated into the ***Intel® PXA27x Processor Family Specification Update*** and are no longer published in other documents.

This document might also contain information that was not previously published.

Affected and Related Documents

Table 1 lists the documents affected by and related to this errata update.

Contact an Intel® representative to obtain the latest revisions of these documents.

Table 1. Affected Documents / Related Documents

Title
Intel® PXA27x Processor Family Developer's Manual
Intel® PXA27x Processor Family Design Guide
Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification

Nomenclature

Errata are design defects or errors. These errata might cause the Intel® PXA27x Processor Family's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor stepping must assume that all errata documented for that stepping are present on all devices unless otherwise noted.

Documentation changes include typos, errors, and omissions from the current published specifications. These changes will be incorporated in the next release of the document.

Specification clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the document.

Specification changes are modifications to the current published specifications. These changes will be incorporated in the next release of the document.

Note: Errata remain in the specification update throughout the product's life cycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and made available upon request. Specification changes, specification clarifications, and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (data sheets, manuals, and so forth).

Intel® PXA27x Processor Family Package Markings

The following figure depicts the location, on specific Intel® PXA27x Processor Family packages, where the actual markings are located. Actual markings are described in Table 2.

Figure 1.Intel® PXA27x Processor Family Package Markings Locations

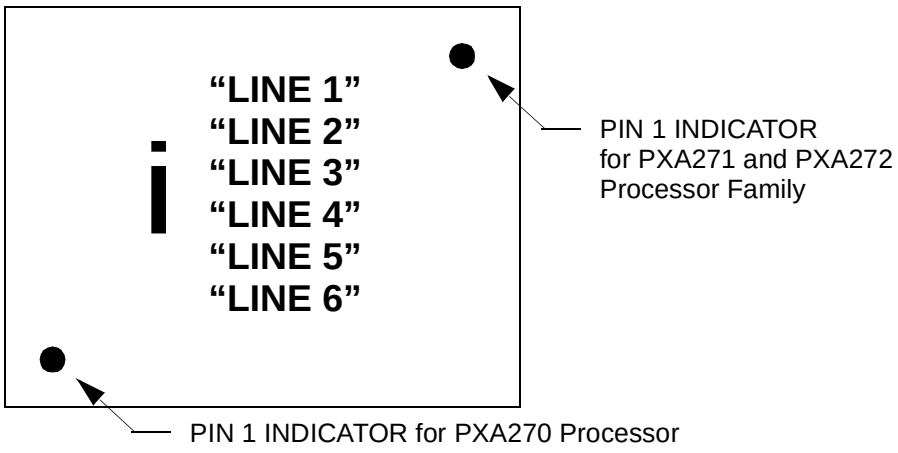

Table 2 describes the actual markings that are on the package at the location indicated by “LINE x” for each package offered.

Note: This table is for example only. It must not be used to determine final production offerings.

Table 2. Example Processor Package Markings (For Example Only)

Discrete/ MCP	13x13 T-PBGA	13x13 T-PBGA	14x14 FFCSP/ 256LTyax/256SD	14x14 FFCSP/ 256LTyax/256SD	14x14 FFSCP/ 2x256L Tyax
Max. Freq.	624 MHz	416 MHz	312 MHz	520 MHz	416 MHz
Lead/ Lead Free	Lead (Pb)	Lead (Pb) Free	Lead (Pb)	Lead (Pb) Free	Lead (Pb) Free
Stepping	C0	C0	C0	C0	C0
Line 1	RCPXA270C0C624	RTPXA270C0C416	LVPXA271FC0312	RVPXA272FC0520	RVPXA272FC0416
Line 2	{FPO#}	{FPO#}	{MCP FPO#}	{MCP FPO#}	{MCP FPO#}
Line 3	{QDF#}	{QDF#}	{QDF#}	{QDF#}	{QDF#}
Line 4	{M} {C} '04	{M} {C} '04	{M} {C} '04	{M} {C} '04	{M} {C} '04
Line 5			ATPO#	ATPO#	ATPO#
Line 6			COO	COO	COO

Summary of Changes

The following tables summarize the errata, specification changes, specification clarifications, and documentation changes that apply to the Intel® PXA27x Processor Family. Intel might fix some of the errata in a future stepping of the component and account for the other outstanding issues through documentation or specification changes as noted. These tables use the following notations:

Notation	Meaning
X	This erratum exists in the stepping indicated. specification change or clarification that applies to this stepping.
(No mark) or (Blank Box)	This erratum is fixed in the listed stepping, or the specification change does not apply to the listed stepping.
Plan Fix	This erratum might be fixed in a future stepping of the product.
Fixed	This erratum has been previously fixed.
No Fix	There are no plans to fix this erratum.
Doc	Intel plans to update the appropriate documentation in a future revision.
No Bug	This erratum has been determined to be a false erratum.
Shaded	This item is either new or modified from the previous version of the document.

Table 3. Summary of Errata (Sheet 1 of 3)

References			C0	Status
Number	Title	Page		
1	CORE: IFU misses an external abort when a lock command is outstanding	13	X	No Fix
2	CORE: Aborted store that hits D-cache might mark write-back data as dirty	13	X	No Fix
3	CORE: Performance Monitor Unit Counts, using performance monitoring event number 0x1, can be incremented erroneously by unrelated core events	14	X	No Fix
4	CORE: In SDS mode, back-to-back memory operations where the first instruction aborts might hang	14	X	No Fix
5	CORE: Lock aborts resulting from I-cache or I-TLB lock operations are not presented properly on the trace interface	14	X	No Fix
6	CORE: CP15 ID register accesses with opcode2 > 0b001 return unpredictable values	15	X	No Fix
7	CORE: Disabling and re-enabling the MMU can hang the core or cause it to execute the wrong code	15	X	No Fix
8	CORE: JTAG parallel register updates require an extra TCK rising edge	16	X	No Fix
9	MMC: SPI mode – if card is deselected, PROG_DONE will not be set	16	X	No Fix
10	MEMC: No MRS command is given when exiting from Alternate Bus Master Mode when SA1111 Address Muxing Mode is enabled	16	X	No Fix

Table 3. Summary of Errata (Sheet 2 of 3)

References			C0	Status
Number	**Title**	**Page**		
11	KBD: Extra Keypad matrix interrupt in IMKP Mode	16	X	No Fix
12	USBH: USBH Register UHCRHPSx[CCS] Bit Set Incorrectly After Power On	17	X	No Fix
13	AC97: Command Done bit remains set after an AC97 cold reset	17	X	No Fix
14	POWER MANAGER: Fast Ramp rates on voltage pins can cause high current consumption.	17	X	No Fix
15	LCD: Reconfiguring the LCD controller retains the previous PPL value for the first line.	18	X	No Fix
16	MSHC: Memory Stick does not come out of SLEEP mode after wake up process.	19	X	No Fix
17	LCD: Overlay1 is not enabled intermittently after re-enabling LCD.	19	X	Plan Fix
18	POWER MANAGER: Processor ignores BATT/VCC faults while exiting sleep mode.	20	X	No Fix
19	KBD: Keyboard Edge-Detect Status Register Incorrect After Standby Mode Wakeup.	20	X	No Fix
20	UART: Character Timeout interrupt remains set under certain software conditions	21	X	No Fix
21	LCD: LCD not enabling in dual panel mode.	21	X	No Fix
22	UDC: UDC does not correctly support alternate interfaces.	22	X	No Fix
23	ICP: Receiver Aborts randomly occur prematurely and without End of frame/Error in FIFO interrupt	23	X	No Fix
24	SSP: OSTimer counter increments incorrectly for SSP Frames in Network mode	23	X	No Fix
25	LCD: Enabling Overlay 2 for YUV420 hangs LCD controller.	23	X	Plan Fix
26	USBOTG: Unable to measure duration of Single-Ended Zero (SE0) for Session Request Protocol (SRP)	24	X	Plan Fix
27	MEMC: Write/Read to/from SDRAM can collide with alternate bus master mode when MDREFR:ALTREFB is set.	25	X	No Fix
28	Power Manger: Core hangs during voltage change when there are outstanding transactions on the bus	25	X	No Fix
29	MMC: MMC unit in SPI mode always waits a minimum of 1 Ncx cycles, even though the MMC spec dictates that SPI mode CMD9 can have a minimum of 0 Ncx cycles.	27	X	Plan Fix
30	SD: SD Controller in SPI mode not receiving data response for CMD9 and CMD10 from some SD Cards	27	X	Plan Fix
31	MEMC: SDRAM Refresh Commands are issued too often during a VLIO access while BREQ is asserted.	28	X	No Fix
32	Interrupt Controller: Unexpected exception vector when ICCR[DIM]=0 and ICMR=0.	28	X	No Fix
33	SSP: TXD line does not tristate when SSP is Slave to Frame	29	X	No Fix
34	PowerManager: Simultaneous BATT and VDD faults results in going to DeepSleep mode twice.	29	X	No Fix
35	CORE: Non-branch instruction in vector table may execute twice after a thumb mode exception	29	X	No Fix
36	UART: UART does not correctly indicate a Framing Error Interrupt in DMA mode.	30	X	No Fix

Table 3. Summary of Errata (Sheet 3 of 3)

References			C0	Status
Number	Title	Page		
37	CLOCKS: System Hangs when enabling RUN/TURBO switching at 520MHz	30	X	No Fix
38	CLOCKS: System Hangs when enabling HalfTurbo Switching	31	X	No Fix
39	MEMC: Memory Controller hangs when entering Self Refresh Mode.	32	X	No Fix
40	SDIO: SDIO Devices Not Working at 19.5 Mbps	32	X	Plan Fix
41	AC97: Command Done bit is never set when data is written in slot12	33	X	No Fix
42	SD/MMC: SD/MMC controller CRC errors with some SD/MMC cards	33	X	Plan Fix
43	USBH: There is no Individual Power Sense Polarity bit for each Host Port. The Power Sense Polarity bit controls the polarity for all three Host Ports.	33	X	Plan Fix
44	KBD: Keypress wakeup from Standby mode is not reliable	34	X	No Fix
45	USBH: USB Host Port 3 in Transceiverless Mode may not work correctly with an external device.	34	X	Plan Fix
46	SD/MMC: SPI mode commands fail even on cards that are compatible with SPI spec 1.0	35	X	Plan Fix
47	CLOCKS AND POWER: PWM Clock Enables do not work as specified	35	X	No Fix
48	ICP: Occasionally EIF, EOF and CRC interrupt are missed when a CRC error is received	36	X	No Fix
49	POWER MANAGER: Batt Fault does not always re-enable GPIO 0 and GPIO 1 as wake-up sources.	36	X	No Fix
50	POWER MANAGER: The processor does not exit from Sleep/ Deep-Sleep Mode.	37	X	Plan Fix
51	SDIO: CMD53 multiple block data transfer with block count set to 0 not supported	37	X	No Fix
52	LCDC: Disable Done Interrupt does not always occur	38	X	No Fix
53	UDC: RCV can not be tied high during UDC transmission when using an external transceiver	38	X	No Fix
54	AC97: AC97 CAR[CAIP] bit field can be incorrectly set	38	X	No Fix

Table 4. Summary of Specification Changes

References			Status
Number	Title	Page	
NOTE: There are no specification changes at this time.			

Table 5. Summary of Specification Clarifications

Number	Document Revision	Page	Status	Specification Clarifications
NOTE: There are no specification clarifications at this time.				

Table 6. Summary of Documentation Changes

Number	Document Revision	Page	Status	Documentation Changes
NOTE: There are no documentation changes at this time.				

Errata

E1. CORE: IFU misses an external abort when a lock command is outstanding.

Problem: A bus abort occurs on a code fetch while an I-TLB lock mcr is outstanding. The IFU fails to abort, instead executing the instruction returned on the aborting transaction. Parity errors are not affected. The bus abort might be due to either an ABORT pin assertion or a multi-bit ECC error on the core. On the core, the bus abort can occur as the result of a master abort, a target abort, or a single data phase disconnect on the system bus.

Implication: TBD

Workaround: Branch flush after every ITLB or I-cache lock.

Status: No Fix

E2. CORE: Aborted store that hits D-cache might mark write-back data as dirty.

Problem: If an aborted store hits clean data in the data cache (data in an aligned four-word range that has not been modified from the core since it was last loaded in from memory or cleaned), the data in the array will not be modified (the store will be blocked), but the dirty bit will be set.

When that line is then aged out of the data cache or explicitly cleaned, the data in that four-word range will be evicted to external memory, even though it has never been changed. In normal operation, this will be nothing more than an extra store on the bus that writes the same data to memory as is already there.

This problem might be visible at the following boundary condition:

1. A cache line is loaded into the cache at address A
2. Another master externally modifies address A
3. An Intel XScale® core store instruction attempts to modify A, hits the cache, aborts because of MMU permissions, and is backed out the of cache. That line should not be marked “dirty”, but because of this defect it will be.
4. The cache line at A then ages out or is explicitly cleaned. The original data from location A will be evicted to external memory, overwriting the data written by the external master.

This situation happen sonly if software is allowing an external master to modify memory that is write-back or write-allocate in the XScale page tables, and depending on the fact that the data is not dirty in the XScale cache to preclude the cached version from overwriting the external memory version. If there are any semaphores or any other handshaking to prevent collisions on shared memory, this should not be a problem.

Implication: TBD

Workaround: For a shared memory region, mark it as write-through memory in the XScale page table to prevent the data from ever being written out as dirty, so the defect does not appear. Alternatively, ensure that any cached copy of the data in external memory is invalidated if an external agent modifies the external copy.

Status: No Fix

E3. CORE: Performance monitor unit counts, using performance monitoring event number 0x1, can be incremented erroneously by unrelated core events.

Problem: The performance monitor unit can be used to count cycles in which the I-cache cannot deliver an instruction. Performance monitoring event number 0x1 is used for this. According to EAS text, the only cycles counted should be those due to an I-cache miss or an I-TLB miss. The following unrelated events in the core also cause the corresponding count to increment when event number 0x1 is being monitored:

- Any architectural event (e.g. IRQ, data abort)
- mstr instructions that alter the CPSR control bits
- Some branch instructions, including indirect branches and those mis-predicted by the BTB
- CP15 mcr instructions to registers 7, 8, 9, or 10 which involve the I-cache or the I-TLB

Each of the items above might cause the performance monitoring count to increment several times. The resulting performance monitoring count might be higher than expected if the above items occur, but never lower. The performance monitor unit uses ic_instValid_qf2h. This signal asserts not just at the proper time, but also due to the events described above.

Implication: TBD

Workaround: There is no way to obtain the correct number of cycles stalled due to I-cache misses and I-TLB misses. One component of the unwanted noise may be filtered out: extra counts due to branch instructions mis-predicted by the BTB. The number of mispredicted branches can also be monitored using performance monitoring event 0x6 during the same time period as event 0x1. The mispredicted branch number can then be subtracted from the I-cache stall number generated by the performance monitor to get a value closer to the correct one. Depending on the nature of the code monitored, this workaround might have limited value.

Status: No Fix

E4. CORE: In SDS mode, back-to-back memory operations where the first instruction aborts might hang.

Problem: If back-to-back memory operations occur in SDS (special debug state, used by ICE and debug vendors) and the first memory operation gets a precise data abort, the first memory operation is correctly cancelled and no abort occurs. However, depending on the timing, the second memory operation might not work correctly. The data cache might internally cancel the second operation, but the register file may have scoreboarded registers for that second memory operation.

The effect is that the part might hang (due to a permanently scoreboarded register) or that a store operation might be cancelled incorrectly.

Implication: TBD

Workaround: In SDS, any memory operation that might cause a precise data abort should be followed by a write-buffer-drain operation. This operation precludes additional memory operations from being in the pipe when the abort occurs. Do not use load multiple/store multiple, which might cause precise data aborts.

Status: No Fix

E5. CORE: Lock aborts resulting from I-cache or I-TLB lock operations are not presented properly on the trace interface.

Problem: This problem affects only processors that use the core's trace interface. An I-cache or I-TLB lock operation that results in lock abort creates a unique internal pipeline signal timing that causes

trouble on the trace interface, which results in the core reporting the event as if it came with the next instruction executed after the aborting lock instruction, the data abort vector.

Implication: TBD

Workaround: None.

Status: No Fix

E6. CORE: CP15 ID register accesses with opcode2 > 0b001 return unpredictable values.

Problem: The XScale core does not implement CP15 ID codes registers other than the Main ID register (opcode2 = 0b000) and the Cache Type register (opcode2 = 0b001). If any of the unimplemented registers are accessed by software (for example, mrc p15, 0, r3, c15, c15, 2), the value of the Main ID register should be returned. Instead, an unpredictable value is returned.

Implication: TBD

Workaround: None.

Status: No Fix

E7. CORE: Disabling and re-enabling the MMU can hang the core or cause it to execute the wrong code.

Problem: If the MMU is disabled via the CP15 control register after being enabled, certain timing cases can cause the processor to hang. In addition, re-enabling the MMU after disabling it can cause the processor to fetch and execute code from the wrong physical address. To avoid these issues, the code sequence below must be employed whenever disabling the MMU or re-enabling it afterwards.

Implication: TBD

Workaround: The following code sequence must be used to disable and/or re-enable the MMU safely. The alignment of the mcr instruction that disables or re-enables the MMU must be controlled carefully, so that it lies in the first word of an instruction cache line:

```
;//@ The following code sequence takes r0 as a parameter. The value of r0 will be
;//@ written to the CP15 control register to either enable or disable the MMU.

    mcr   p15, 0, r0, c10, c4, 1          ;// @ unlock I-TLB
    mcr   p15, 0, r0, c8, c5, 0           ;//@ invalidate I-TLB
    mrc   p15, 0, r0, c2, c0, 0           ;//@ CPWAIT
    mov   r0, r0
    sub   pc, pc, #4
    b     AfterAlign                      ;//@ branch to aligned code
    ALIGN 32                              ;//@ align to 32 bytes
AfterAlign
    mcr   p15, 0, r0, c1, c0, 0               ;//@ enable/disable MMU, caches
    mrc   p15, 0, r0, c2, c0, 0               ;//@ CPWAIT
    mov   r0, r0
    sub   pc, pc, #4
```

Status: No Fix

E8. CORE: JTAG parallel register updates require an extra TCK rising edge.

Problem: The IEEE 1149.1 spec states that the effects of updating all parallel JTAG registers should be seen on the falling edge of TCK in the Update-DR state. The XScale core parallel JTAG registers incorrectly require an extra TCK rising edge to make the update visible. Therefore, operations like hold-reset, JTAG break, and vector traps require either an extra TCK cycle by going to run-test-idle or by cycling through the state machine again to trigger the expected hardware behavior.

Implication: TBD

Workaround: If the JTAG interface is polled continuously, this erratum has no effect. If not, an extra TCK cycle can be caused by going to run-test-idle after writing a parallel JTAG register.

Status: No Fix

E9. MMC: SPI mode – if card is deselected, PROG_DONE will not be set.

Problem: If changing SPI chip selects, the PROG_DONE bit does not get updated with the state of the selected card.

Implication: If programming card0, then switch to card1, then come back to card0, there is no way of knowing if card0 ever finished programming.

Workaround: User can switch the MMDAT signal functionality to GPIO functionality and monitor the signal by reading the GPIO Status register until the signal goes high.

Status: No Fix

E10. MEMC: No MRS command is given when exiting from alternate bus master mode when SA-1111 address muxing mode is enabled.

Problem: When using alternate bus master with MDCNFG[SA1110x] set, after the alternate bus master has given the bus back to the processor, if the first access by the processor is a VLIO access, the MRS commands to switch the SDRAMs back into burst-of-4 mode occurs after a long delay. If SDRAM is accessed before the MRS command is performed, data returned to the processor registers will not be valid.

Implication: TBD

Workaround: Use any of the following options to workaround this erratum:

- Do not use SA-1111 address muxing mode with an alternate bus master.
- The alternate bus master can perform the MRS command to put the SDRAM into burst-of-4 mode before de-asserting the MBREQ signal.

Status: No Fix

E11. KBD: Extra keypad matrix interrupt in IMKP mode.

Problem: An unexpected interrupt during keypad-matrix manual scan in ignore-multiple-keypress mode (IMKP). The test consists of pressing one, single, valid key and holding it down, then pressing a second, single, valid key and holding it down, then both pressed keys are released simultaneously. This test results in three vectors to the interrupt-service routine where two are expected.

The first vector to the ISR occurs on first key down. The second vector is unexpected. This is possibly caused by the second key down. The third vector occurs approximately at the time of all keys up.

Debounce interval = 3ms, KPC[IMKP] = 1, KPC[ASACT] = 0; KPC[MI] = 1.

Implication: TBD

Workaround: Do not set KPKDI[Interval] to less than 10ms.

Status: No Fix

E12. USBH: USBH register UHCRHPSx[CCS] bit set incorrectly after power on

Problem: The UHCRHPSx[CCS] bit may be set after power on reset. Under normal conditions, when set, this bit indicates that a USB device is connected to the USB host port. The error is that this bit is being set even though there is no device physically connected to the USB host unit.

Implication: TBD

Workaround: Disable global port power for the USB host unit. Only enable individual port power for the USB host ports that will be used in the system.

Status: No Fix

E13. AC97: Command-done bit remains set after an AC97 cold reset.

Problem: After an AC97 cold reset is asserted (GCR[nCRST]=0) when the AC97 controller unit is enabled again, the command-done bit (GSR[CDONE]) is getting set when it should not.

1. Enable the AC97 unit.
2. Enable the codec.
3. Write some data to the codec register using AClink.
4. Wait for command-done bit (GSR[CDONE]) to set and then clear it.
5. Shut down the codec by writing 0x1000 to codec address 0x26.
6. Wait for command-done bit (GSR[CDONE]) to set and then clear it.
7. Generate wakeup event from Codec and wait for primary codec resume bit (GSR[PRESINT]) or secondary codec resume bit (GSR[SRESINT]) to set.
8. Once resume bit is set, assert a cold reset by writing 0 to GCR[nCRST].
9. Enable the AC97 controller and read the GSR. The command-done bit is set; however, it should be clear.

Implication: TBD

Workaround: After asserting cold reset, the GSR[CDONE] bit must be cleared by writing a 1 to GSR[CDONE].

Status: No Fix

E14. POWER MANAGER: Fast ramp rates on voltage pins can cause high current consumption.

Problem: If voltage rise/stabilization time is too fast for any of the voltage pins (VCC_core, VCC_mem, VCC_batt, etc.), then the internal circuitry causes high current consumption on these voltage pins.

This condition will not damage the internal circuitry of the processor. The duration of the high current consumption is less than 20µsec.

The following describes the current consumption in order of voltage domain bootup sequence.

- If the voltage rise/stabilization time of VCC_BATT is longer than 500µsec at room temperature, then the combined current on VCC_BATT will be less than 100mA for less than 20µsec.

- If the voltage rise/stabilization time of VCC_USB, VCC_IO, VCC_MEM, VCC_BB, VCC_LCD, VCC_USIM is longer than 500µsec at room temperature, then the combined current on VCC_USB, VCC_IO, VCC_MEM, VCC_BB, VCC_LCD, VCC_USIM will be less than 500mA for less than 20µsec.
- If the voltage rise/stabilization time of VCC_CORE, VCC_SRAM, VCC_PLL is longer than 500µsec at room temperature, then the combined current on VCC_CORE, VCC_SRAM, VCC_PLL will be less than 100mA for less than 20µsec.

Example of battery life consumption:

— Assume square current spike = 850mA = 0.85Coulomb/sec

— Current drain of the current spike lasting 100µsec

— 0.85 Coulomb/sec * 0.0001 sec = 85 µCoulomb total drain

— Assume 150mA-hr battery = 540 Coulombs

— Calculate current spike percentage of total battery capacity: 85µC / 540C = 1.57 e-7

— Current spike consumes only 0.0000157% of battery capacity.

Implication: High current consumption on processor voltage supply pins can cause external power management circuitry to enter into an error/shutdown condition.

Workaround: Voltage rise/stabilization time must be determined by the system designer to mitigate any error condition on external circuitry.

Status: No Fix

E15. LCD: Reconfiguring the LCD controller retains the previous PPL value for the first line.

Problem: If you program the LCD controller for one configuration, then disable the LCD controller and program it for another configuration and enable it again, the first line retains the pixels-per-line (PPL) value from the previous configuration.

For example, configuring the LCD to 320x200 and then disabling it and reconfiguring it to 640x480, and enabling it again, the first line retains the PPL value from the previous configuration, which is 320.

This issue could effect more than just the first frame, due to the fact that the LCD DMA is initialized with a size to fit the frame size. Data from the first frame could spill over into the second frame, and so on.

Implication: If the wrong number of pixels are sent to the LCD panel for the first line of a frame, the display on the LCD screen could be corrupt.

Workaround: When configuring the LCD the second time, configure it to the desired LCD configuration, enable the LCD, disable it, and then enable it again, for the third time. This ensures that when the LCD controller is enabled for the third time, it will begin the frame with the right PPL value.

Status: No Fix

E16. MSHC: Memory Stick does not come out of SLEEP mode after wake up process.

Problem: The test is trying to put the memory stick to ”SLEEP” and then wake it up by sending the “RESET” TPC. The manual says that “Memory Stick wakes up and performs a packet processing, when a packet transfer by SET_CMD TPC or Write_Reg occurs at SL = 1 in Status Register 0”.

The memory stick does not wake up after sending a SET_CMD TPC or Write_Reg once. However, it does wake up on sending SET_CMD TPC or Write_Reg twice. After reading the value of the MSINT register (address 0x4180_0008)after sending the first TPC, it contains the value 0x0000200D, indicating that a time out error occurred.

After reading the “SLEEP” bit in the “status register 0”, it remains “1” after the first TPC and changes to “0” after the second TPC, indicating it was asleep after the first TPC transmission and awake after the second TPC transmission.

How to reproduce:

- Send “SLEEP” SET_CMD TPC.
- Read the Status Register 0 (Sleep bit should be 1).
- Wake up the Memory Stick by sending “RESET” SET_CMD TPC.
- Pause for 10 seconds.
- Read the Status Register 0 (Sleep bit should be 0 but it shows 1).
- Send “RESET” again.
- Read the Status Register 0 (Sleep bit shows 0).

Implication: TBD

Workaround: Software should send two reset TPC commands to the Memory Stick. Memory Stick wakes up after two reset TPCs.

Status: No Fix

E17. LCD: Overlay1 is not enabled intermittently after re-enabling LCD.

Problem: When enabling and disabling LCD several times, intermittently, Overlay1 output is all zeros instead of intended frame buffer overlay pattern. Overlay2 works fine.

Implication: TBD

Workaround: Option1:

Users must not use Overlay 1 but can use Overlay 2.

Option2: Users can use Overlay 1 by:

1. Programming the LCD registers
2. Enabling the LCD (LCCR0[ENB] = 1)
3. Disabling the LCD (LCCR0[DIS] = 1)
4. Reprogramming the lcd registers but not the overlay1/cursor registers. Also, do not write DMA FDADR registers
5. Enabling the LCD (LCCR0[ENB] = 1)
6. Performing a quick disable of the LCD (LCCR0[ENB] = 0)
7. Programming the DMA registers
8. Enabling the LCD (LCCR0[ENB] = 1)
9. Updating the overlay/cursor registers if required. Follow the instructions mentioned in the manual in section 7.4.7 for overlay1 and 7.4.10 for cursor as to how to change the overlay/ cursor registers dynamically when the LCD is enabled.

Status: Plan Fix

E18. POWER MANAGER: Processor ignores BATT/VCC faults while exiting sleep mode.

Problem: When asserting nBATT_FAULT or nVCC_FAULT during Sleep mode with the corresponding IDAE set, and the fault is still present after boot, the part does not enter Deep Sleep. However, if the IDAE bit is clear, and the fault is still present after boot, the part enters Deep Sleep as expected.

Implication: If a FAULT continues to occur while the processor is exiting Sleep Mode, then the processor is going to continue to bootup and consume power. There must be enough battery power or the system must be able to tolerate the fault condition until the workaround below is complete.

Workaround: Before entering sleep or deep sleep via software, first write 1 to the PMCR[IAS] bit to enable interrupts and then a separate write must be performed to set the desired PMCR[xIDAE] bits. When the chip exits sleep mode, the interrupt pertaining to Power Management Unit must be unmasked, ICMR[PWR_I2C] = 0b1, to find out if a fault has occurred. Alternatively, the PMCR[INTRS] bit can be checked.

Status: No Fix

E19. KBD: Keyboard Edge-Detect Status Register Incorrect After Standby Mode Wakeup.

Problem: When the processor is in standby mode and it is programmed to wake up by a KeyPad GPIO pulse, the PKSR (Keyboard Edge detect Status) does not correctly report the GPIO that caused the Wakeup.

Scenario 1 (Correct):

If only one keypad GPIO is pulsed, and it is programmed to wakeup from standby in PKWR, the PKSR reports the correct GPIO as the source of the wakeup as expected.

Scenario 2 (Incorrect):

If ALL keypad GPIOs are pulsed but only one is programmed in PKWR to wake up, the PKSR reports all the GPIOs (except for the ones in use) as the source of the wakeup. This is wrong, since only one GPIO was programmed to wake up, regardless of the fact that all GPIOs were pulsed.

Therefore, the following documentation will be added to the Developer’s Manual:

"If any of the PKWR bits are enabled, then only the keypad related to that PKWR will cause a wakeup. If there is any other activity (active high) on any other keypad pins at this time, then irrespective of its PKWR setting, it will be registered into the PKSR."

Implication: The user will not be able to detect which keypress woke the processor out of Standby mode.

Workaround: The workaround is to only use PKSR as an indicator that a keypress woke the processor. The user will not be able to tell which key woke the processor.

Status: No Fix

E20. UART: Character Timeout interrupt remains set under certain software conditions

Problem: The issue is that randomly the character timeout interrupt does not clear and the DR bit is not set. The failure was reproduced by adding a software delay loop inside the character timing interrupt routine between reads from the FIFO. The test continuously repeats and increases the software delay. After a few iterations, the tests get into a continuous interrupt loop, where the character timeout interrupt is set, but there is not any data in the FIFO. If the delay loop is placed just outside

the loop to read out the data, the test never fails. So the issue looks to be related to the amount of time between reading data out of the FIFO.

Pseudo Code for the Character Timeout Interrupt Handler routine test:

1. Read LSR and check for errors
2. Read Data from FIFO
3. Software Delay (Delay increases after each test)
4. Read LSR, check for errors, and LOOP back to (2) if DR is SET.
5. DONE

if step (3) is placed in front of step (1) the issue never occurs.

Implication: The software servicing the UART can be trapped in an infinite loop.

Workaround: Steps 2 and 6 have been added to the Interrupt Handler routine. Disabling the Receiver Time-Out interrupt via IER[RTOIE] (step 2) prevents the Receiver Time-Out interrupt from becoming continuously set.

1. Read the Line Status Register (LSR) and check for errors
2. Disable the Receiver Time-Out interrupt via IER[RTOIE]
3. Read Data from the UART FIFO
4. Software Delay (Increments after each test)
5. Read LSR, check for errors, and LOOP back to (2) if DR is SET.
6. No more data in FIFO: Re-enable RTO interrupt via IER[4]
7. DONE

Therefore, users must disable the Receiver Time-Out interrupt, then read data from the FIFO, then re-enable the Receiver Time-Out interrupt, before exiting the handler routine.

Status: No Fix

E21. LCD: LCD not enabling in dual panel mode.

Problem: After the LCD unit is dynamically changed from single panel mode to dual panel mode, the LCD unit will not output any data on the LCD pins.

Implication: TBD

Workaround: Do the following sequence to change from single panel mode to dual panel mode:

Configure the LCD to Single panel mode
Enable the LCD
Disable the LCD
Reconfigure the LCD to Dual Panel mode without programming the DMA registers.
Enable the LCD
Do a quick disable (not normal disable).
Program the LCD DMA registers
Enable the LCD for operation in Dual Panel mode.

Status: No Fix

E22. UDC: UDC does not correctly support alternate interfaces.

Problem: When a SET_INTERFACE command is sent from the host, the UDC reallocates the internal FIFO memory.

When USB Client comes up after the USB Reset and Set Configuration is done other than Configuration 0, it does not allow access to all the Interfaces available for that configuration.

Some excerpts of USB Specification describing use of Interfaces

---

As part of the configuration process, the host sets the device configuration and, where necessary, selects the appropriate alternate settings for the interfaces.

Interfaces are numbered from zero to one less than the number of concurrent interfaces supported by the configuration. Alternate settings range from zero to one less than the number of alternate settings for a specific interface. The default setting when a device is initially configured is alternate setting zero.

see section: Set Interface

This request allows the host to select an alternate setting for the specified interface. This request cannot be used to change the set of configured interfaces (the SetConfiguration() request must be used instead).

see section: Configuration

The descriptor describes the number of interfaces provided by the configuration. Each interface may operate independently. For example, an ISDN device might be configured with two interfaces, each providing 64 Kb/s bi-directional channels that have separate data sources or sinks on the host. Another configuration might present the ISDN device as a single interface, bonding the two channels into one 128 Kb/s bi-directional channel.

An endpoint is not shared among interfaces within a single configuration unless the endpoint is used by alternate settings of the same interface. Endpoints may be shared among interfaces that are part of different configurations without this restriction.

Implication: The processor USB Client does not follow the USB Specification when dealing with different interfaces. External USB host connected to the processor USB Client cannot switch correctly between alternate interfaces on the processor USB Client. This issue makes the processor USB Client difficult to be a compound device.

Workaround: TBD

Status: No Fix

E23. ICP: Receiver Aborts randomly occur prematurely and without End of frame/Error in FIFO interrupt

Problem: Randomly, Receiver Abort (ICSR0[RAB]=1) occurs prematurely and without the EOF status/ interrupt bit set (ICSR1[EOF]=1). When this event occurs, there is always 2 bytes left in the RX FIFO.

A Receiver Abort should always cause an EOF, but in the failing cases, only the Receiver Abort is set.

Adding ~15us delay to the receive data available interrupt (IIR[IID]=0b10) service routine causes this failing case to pass, but limits the transfer size to the size of the FIFO since the added delay will cause an overrun condition.

Implication: TBD

Workaround: Software must do retries after RAB has occurred. Once a RAB event occurs, continue to read data out of the FIFO until the EOF flag sets, then throw away all the data of that frame, and then make a request to the host to resend/retry the last frame of data again.

Status: No Fix

E24. SSP: OSTimer counter increments incorrectly for SSP Frames in Network mode

Problem: When SSP unit is in network mode, when PSP protocol is used, and the SSP is master to Frame (irrespective of sspsclk direction) and FSRT bit is set to 1, OStimer frame counter increments more than the frame is asserted by SSP. For example, 11 frames are asserted as seen on a logic analyzer, but OStimer is showing that the frame signal is asserted for 12 times. OS timer works fine when FSRT is cleared and when in non-network mode.

Implication: TBD

Workaround: Subtract one from the value that is read from the OSTimer frame counter.

Status: No Fix

E25. LCD: Enabling Overlay 2 for YUV420 hangs LCD controller.

Problem: Enabling Overlay 2 in YUV 420 mode causes the LCD controller to stop operating (DMA activity stops and the screen fades away). The test is unable to gracefully shutdown the LCD controller after this failure.

Enabling Overlay 2 in RGB mode has no problems.

Implication: TBD

Workaround: To Enable/Disable Overlay 2 in 4:2:0 mode:

Step 1: a. Enable the LCD with Overlay2 disabled.

Step 2: a. Enable Overlay 2 in RGB mode with minimum size possible so that only one frame worth of data fit in Channel 2 FIFO. Size of Channel 2 FIFO is 128 bytes. Program the Descriptors and write to O2CR1.

b. Run at least 1 frame with Overlay 2.

c. Disable the Overlay 2 by writing to O2CR1 and Frame Branch Registers (FBRx).

d. Wait for 3 base EOF interrupts.

Step 3: a. Enable Overlay 2 in YUV 420 mode. Write to O2CR1.

b. Unmask/Clear the input underrun for Channel 2. Wait for input underrun from Channel 2. Write DMA descriptors for Channels 2, 3, and 4.

Step 4: a. Now Overlay 2 in YUV 420 mode is enabled and running.

Step 5: a. If Overlay 2 needs to be disabled, Disable the Overlay 2 by writing to O2CR1 and Frame Branch Registers (FBRx).

b. If the Overlay 2 needs to be enabled again, Go to Step 3.

Step 6: a. If LCD needs to disabled, disable the Overlay 2 as mentioned in Step 4, then Disable LCD, and go to Step 1.

Pseudo Code to Enable RGB Mode:

```
Write ovl2c2
Write ovl2c1 (O2EN = 1)
Write fdadr2
```

Pseudo Code to Disable RGB Mode:

```
Write ovl2c1 (O2EN = 0)
Clear LCSR1 BS2 bit (LCSR1 = 0x200
Write fbr2 (BRA=BRANCH_AFTER_CURRENT_FRAME, BINT=SET_SR_BS_BIT)
Wait for branch to complete (LCSR1 == 0x200)
```

Pseudo Code to Enable 4:2:0 mode:

```
Write ovl2c2
Write ovl2c1 (O2EN = 1)
Clear LCSR1 IU2 bit (LCSR1 = 0x02000000);
Wait for under-run LCSR1 = 0x02000000
Write fdadr2
Write fdadr3
Write fdadr4
```

Pseudo Code to Disable 4:2:0 mode:

```
Write ovl2c1 (O2EN = 0)
Clear LCSR1 BS2 bit (LCSR = 0xe00)
Write fbr2 (BRA=BRANCH_AFTER_CURRENT_FRAME BINT=SET_SR_BS_BIT)
Write fbr3 (BRA=BRANCH_AFTER_CURRENT_FRAME BINT=SET_SR_BS_BIT)
Write fbr4 (BRA=BRANCH_AFTER_CURRENT_FRAME BINT=SET_SR_BS_BIT)
Wait for branch to complete (LCSR1=0xe00)
```

Status: Plan Fix

E26. USBOTG: Unable to measure duration of Single-Ended Zero (SE0) for Session Request Protocol (SRP)

Problem: According to On-the-Go Supplement to the USB 2.0 Specification, we must be able to detect that a Single-Ended Zero (SE0) condition is driven on the USB bus for at least 2 ms before we can initiate a Session Request Protocol (SRP). See paragraph below.

Sec. 5.3.2..."A second initial condition for starting a new session is that the B-device must detect that both the D+ and D- data lines must have been low (SE0) for at least 2 ms (TB_SE0_SRP min.). This ensures that the A-device has detected a disconnect condition from the device."...

Software has the ability, through reset interrupt, to see when the SE0 state was entered, but it has no way to determine how long it has been in this state, to determine if the 2 ms requirement has been met.

Implication: Dual role devices (A and B devices) are required to be able to initiate and respond to an SRP, therefore, there is the potential that the USB OTG will not pass a USB OTG compliance test.

Workaround: The USB OTG specification says, in section 5.3.2:

"A second initial condition for starting a new session is that the B-device must detect that both the D+ and D- data lines must have been low (SE0) for at least TB_SE0_SRP min. This ensures that the A-device has detected a disconnect condition from the device." Currently, TB_SE0_SRP is specified at 2 msec minimum.

The workaround is to require VBUS to be low for at least 2 msec, and assume that the data lines are low, if VBUS is low. This assumption would work for devices that make the same assumption, but nothing in the specification requires this.

Status: Plan Fix

E27. MEMC: Write/Read to/from SDRAM can collide with alternate bus master mode when MDREFR:ALTREFB is set.

Problem: When ALTREFB is programmed to 1 and an alternate bus master requests the bus, an SDRAM access in progress may not be allowed to finish before the bus is released. This issue is found in both fly-by mode and non-fly-by mode.

Currently, there is a provision in the specification that ALTREFA and ALTREFB cannot both be set at the same time. Given the workaround below, this will only allow the refreshes to occur both before and after an alternate bus master, or only before an alternate bus master.

Implication: This will potentially cause more refreshes than is necessary according to the SDRAM specification.

Workaround: Never set MDREFR[ALTREFB]. This will cause a refresh cycle to always occur before allowing an alternate bus master to be granted the bus.

Status: No Fix

E28. Power Manger: Core hangs during voltage change when there are outstanding transactions on the bus

Problem: If coprocessor 14 register 7 is written for a voltage change sequence (PWRMODE[VC] = 0b1) and there are outstanding core transactions on the internal bus, the core hangs.

Implication: Unpredictable results can occur if the core hangs.

Workaround: The workaround is to make sure all core transactions are complete and no new core transactions will attempt to get on the internal bus, then initiate a voltage change by setting PWRMODE[VC], then waiting until the PWRMODE[VC] bit is cleared. The VC bit will be cleared within 50ns of the voltage change initiation. If software requires the entire voltage change sequence to complete, i.e. no more communication with the external power manager IC, the software must wait until the PVCR[VCSA] bit is clear.

The following workaround has been tested and will handle all cases.

```
@ WORKAROUND - Core hangs on voltage change at different
@ alignments and at different core clock frequencies
@ To ensure that no external fetches occur, we want to store the next
@ several instructions that occur after the voltage change inside
@ the cache. The load dependency stall near the retry label ensures
@ that any outstanding instruction cacheline loads are complete before
@ the mcr instruction is executed on the 2nd pass. This procedure
@ ensures us that the internal bus will not be busy.

@ -- Begin EnterVoltageChange__asm (void)
    .section.text
    .global EnterVoltageChange__asm
    .align 7                          @ align code to fit in one page
EnterVoltageChange__asm:
    stmfd   sp!, {r4}
    mrs     r0, CPSR                  @ disable interrupts
    mov     r4, r0
    orr     r0, r0, #0xC0
    msr     CPSR_c, r0

    ldr     r0, =0x41300000           @ APB register read and compare
    ldr     r0, [r0]                  @ fence for pending slow apb reads
    cmp     r0, #0

    mov     r0, #8                    @ VC bit for PWRMODE
    movs    r1, #1                    @ don't execute mcr on 1st pass
    mov     r2, #0x0A000000           @ uncacheable memory to force memory read
                                  @ CHANGE THIS ADDRESS for your environment
                                  @ MUST be mapped into pagetable prior to
                                  @ this function call
retry:
    ldreq   r3, [r2]                  @ only stall on the 2nd pass
    cmpeq   r3, #0                    @ compare causes fence on memory transfers
    cmp     r1, #0                    @ is this the 2nd pass?
    mcreq   p14, 0, r0, c7, c0, 0     @ write to PWRMODE on 2nd pass only

@ Read the VC bit until it is 0, indicates that the VoltageChange is done.
@ On the first pass, we never set the VC bit, so it will be clear already.
VoltageChange_loop:
    mrc     p14, 0, r3, c7, c0, 0
    tst     r3, #0x8
    bne     VoltageChange_loop
```

```
    subs    r1, r1, #1            @ update conditional execution counter
    beq     retry

    msr     CPSR_c, r4               @ restore interrupts to original state

    ldmfd   sp!, {r4}
    mov     pc, lr

    .data
@ -- End EnterVoltageChange__asm
```

Status: No Fix

E29. MMC: MMC unit in SPI mode always waits a minimum of 1 Ncx cycles, even though the MMC spec dictates that SPI mode CMD9 can have a minimum of 0 Ncx cycles.

Problem: The MMC unit is not compliant with the MMC Spec 3.2 in SPI Mode.

Version 2.11 of the MMC Spec, in SPI mode, requires 1 or more Ncr cycles between the end of the response and the beginning of the data block for CMD9 (SEND_CSD command).

Version 3.1 and 3.2 of the MMC Spec, in SPI mode, requires 0 or more Ncx cycles between the end of the response and the beginning of the data block for CMD9 (SEND_CSD command).

The MMC unit always waits a minimum of 1 Ncx cycle between the response and the data block, therefore, MMC unit is not compliant with the version 3.2 of the Spec.

Implication: TBD

Workaround: TBD

Status: Plan Fix

E30. SD: SD Controller in SPI mode not receiving data response for CMD9 and CMD10 from some SD Cards

Problem: The SD unit is not compliant with SD Spec 1.01 in SPI Mode.

Version 1.0 of the SD Spec, in SPI mode, requires 1 or more Ncr cycles between the end of the response and the beginning of the data block for CMD9 (SEND_CSD command).

Version 1.01 of the SD Spec, in SPI mode, requires 0 or more Ncx cycles between the end of the response and the beginning of the data block for CMD9 (SEND_CSD command).

The SD unit always waits a minimum of 1 Ncx cycle between the response and the data block, therefore, the SD unit is not compliant with the 1.01 Spec.

Implication: TBD

Workaround: TBD

Status: Plan Fix

E31. MEMC: SDRAM Refresh Commands are issued too often during a VLIO access while BREQ is asserted.

Problem: During a VLIO access, if BREQ is asserted, the processor issues Refresh Commands to SDRAM too often. This occurs after BREQ is asserted and before BGNT gets asserted. This situation lasts for as long as VLIO CS is asserted, which can be several microseconds. During this time, the processor is refreshing SDRAM at 100ns interval although programmed interval is around 8usec (80x difference). Everything goes back to normal after BGNT is released.

Implication: TBD

Workaround: TBD

Status: No Fix

E32. Interrupt Controller: Unexpected exception vector when ICCR[DIM]=0 and ICMR=0.

Problem: When ICCR.DIM = 0 and ICMR = 0 and a direct key press is performed, the Interrupt Control IRQ Pending (ICIP) register, does not indicate a pending interrupt after coming up from idle, but the core attempts to vector to an exception vector address (0x000 to 0x01C).

If ICMR=0x0 and ICCR[DIM]=0b0 and a direct key press is performed, then the correct behavior should be:

After the key is pressed, the processor wakes from IDLE, ICPR gets updated, the ICIP (or ICFP) does not get updated, and the processor should not vector to any vector address (0x00 to 0x1C).

The incorrect behavior is:

After the key is pressed, the processor wakes from IDLE, ICPR gets updated, the ICIP (or ICFP) does not get updated, but the processor tries to vector to 0x18 (for IRQ handler) or 0x1C (for FIQ handler), based on the contents of ICLR.

Keypad and DIM Bit and ICMR test failure:

ICCR = 0x00000000, ICMR = 0x04000000.

KPC = 0x000001C3; KPC[DIE] = 1; KPC[DE] = 1;

KPKDI[DIRECT KEY DEBOUNCE INTERVAL] = 20 (msec).

Implication: TBD

Workaround: Code below disables interrupts to the core before going to Idle and re-enables core interrupts after returning from Idle.

```
IDLE:
.global IDLE
    mrs r0, cpsr              @ read current processor status register
    orr r0, r0, #0xC0         @ disable core interrupts
    msr cpsr_c, r0            @ update the current processor status register
    mov r0, #1
    mcr p14, 0, r0, c7, c0, 0 @ Set IDLE
    mrc p14, 0, r0, c7, c0, 0 @ CPWAIT ROUTINE
```

```
mov r1, r1                @ CPWAIT ROUTINE
sub pc, pc, #4            @ CPWAIT ROUTINE
mrs r0, cpsr              @ read current processor status register
bic r0, r0, #0xC0         @ enable core interrupts
msr cpsr_c, r0            @ update the current processor status register
mov pc, r14               @ RETURN
```

Status: No Fix

E33. SSP: TXD line does not tristate when SSP is Slave to Frame

Problem: When the SSP is a Slave to Frame (SFRMDIR=1) and Master of Clock (SCLKDIR=0) using the TI SSP Format, the SSP will not tri-state the TXD line at the end of the last frame when the TTE=1 and TTELP=1 and the SCR>3.

When TI protocol is used, the SSP unit is Slave to Frame and Master/Slave to clock, TTE=1, TTELP = 1, Data Size = 4.

When a data value of 0xA is transmitted, the last data bit is zero, and when a pullup is applied to the TXD pin, the line should go high, but it always remains low.

Similarly, when a data value of 0x5 is transmitted, the last data bit is one, and when a pull-down is applied to the TXD pin, the line should go low, but it always remains high.

When TTE=1, TTELP = 0 is used, the TXD line will tristate at the beginning and end of transfer. Also, the TXD line works fine when the SSP is Master to Frame.

Implication: TBD

Workaround: None.

Status: No Fix

E34. PowerManager: Simultaneous BATT and VDD faults results in going to DeepSleep mode twice.

Problem: When in any mode, asserting simultaneous BATT_FAULT and VDD_FAULT signals, and while BIDAE is not equal to VIDAE, requires the user to go to DeepSleep twice, once automatically due to one of the xIDAE bits being clear, and once via software due to one of the xIDAE bits being set.

Another observation is that a very short (approx. 332usec) nRESET_OUT assertion occurs followed by a bootup (with no change to RCSR).

Implication: TBD

Workaround: PMCR[BIDAE] and PMCR[VIDAE] must be identical.

Status: No Fix

E35. CORE: Non-branch instruction in vector table may execute twice after a thumb mode exception

Problem: If an exception occurs in thumb mode and a non-branch instruction is executed at the corresponding exception vector, that instruction may execute twice. Typically, instructions located at exception vectors must be branch instructions which go to the appropriate handler, but the ARM architecture allows the FIQ handler to be placed directly at the FIQ vector (0x0000001c/ 0xffff001c) without requiring a branch.

Implication: The first instruction in thumb mode of a FIQ handler may be executed twice if it is not a branch instruction.

Workaround: If a no-op is placed at the beginning of the FIQ handler, the no-op will execute twice and no incorrect behavior will result. If a branch instruction is placed at the beginning of the handler, it will not be executed twice.

Status: No Fix

E36. UART: UART does not correctly indicate a Framing Error Interrupt in DMA mode.

Problem: When DMA is enabled and a "Framing Error" occurs, the UART generates an interrupt, but the IIR register does not show any pending interrupts (IIR[nIP]=0b1).

Reading the registers does not clear the interrupt. The UART will continuously interrupt the CORE.

If the RX FIFO is emptied until there is only 4 bytes of data left in the FIFO, the UART will finally generate a RLS interrupt showing a "framing" error.

The UART handles errors differently in DMA mode. When an error occurs the error gets tagged in the rx fifo with the erroneous byte. In interrupt mode, the error is flagged (processor interrupted) when the erroneous byte is read out of the bottom of the fifo. In DMA mode, the error is flagged as soon as it is detected. In DMA mode, the UART correctly interrupts the processor when the error is first detected but it does not update the IIR register until the error is read out of the FIFO. Thus, the processor is interrupted with no interrupt pending. The correct operation is to both interrupt the processor and update the IIR register when the error is detected.

Implication: The UART generates an interrupt, but the IIR register does not show any pending interrupts.

Workaround: If an interrupt is generated, but the IIR register does not show any pending interrupts, then read data out of the RX FIFO until either it is empty or a valid interrupt is generated. The error interrupt to the core will not clear until the erroneous byte is read out of the RX FIFO.

Status: No Fix

E37. CLOCKS: System Hangs when enabling RUN/TURBO switching at 520MHz

Problem: The system will hang when performing switching from Run Mode frequency to Turbo Mode frequency.

The test flow is as follows:

- Bootup
- Perform a Frequency / Voltage change to L=16, N=2.5, A=1, B=1, K0DB4=1, K1DB2=1,K2DB2=1. This gives core = 208MHz (run mode), Sys Bus = 208MHz, and MEMC = 208MHz. Note the Turbo bit (T) is not set yet.
- Display a code of “0xFFAA0000”
- Go to Turbo mode by setting the Turbo bit.
- Display a code of “0xFFBB0000”

This test intermittently hangs after the “FFAA” code and before the “FFBB” code. Intermittently means that the test can switch back and forth between Turbo mode and Run mode and a failure will most likely occur between 10,000 iterations and 400,000 iterations.

Implication: TBD

Workaround: Workaround for going from RUN (208MHz) to TURBO (520MHz):

```
    Set CCCR[CPDIS]                  // Disable the Core PLL
    Clear CCCR[PPDIS]                // Enable the Peripheral PLL
    Set CLKCFG[F]                    // Initiated a Frequency Change
// Core freq = 13MHz, Peripherals = normal freq (PPLL = 312MHz)
    Set CCCR[PLL_Early_EN]           // Allow the Core PLL to ramp up early
    Wait for CCSR[CPLCK] to set and CCSR[PPLCK] to set.
    Clear CCCR[CPDIS]                // Enable the Core PLL
    Set CLKCFG[F] and Set CLKCFG[T]  // Initiate a Frequency Change
// Core is now running in Turbo mode @ 520Mhz
```

No Workaround needed for switching from TURBO (520MHz) to RUN (208MHz). Switching from TURBO to RUN works fine by clearing CLKCFG[T].

Estimated Latency Timings:

From RUN to 13M mode is about 45us

From 13M to PLL lock is about 15us

From PLL lock to TURBO is about 100us

Total latency is about 160us, where 15us of the time you can execute code while the core is operating in 13MHz mode. This time can increase if the PLL lock time changes.

If you choose to go from 208MHz RUN mode, into a different product point, such as 416MHz Turbo, then switch into 520MHz Turbo, the latency is about 230us.

Status: No Fix

E38. CLOCKS: System Hangs when enabling HalfTurbo Switching

Problem: The system will hang when switching into and out of HalfTurbo, i.e. when going from Run mode to HalfTurbo mode and from Turbo mode to HalfTurbo mode. The failure does not occur when only switching from Run mode to Turbo mode and back.

This failure occurs on, but not limited to, the L=8, N=3, T=1, HT=1, A=1, B=1 product point (i.e. Core = 156MHz, System Bus = 104MHz, MEMC = 104MHz)

Failures on this product point are intermittent. Intermittent means that on powerup, the test will either run or hang. If the first change to HalfTurbo works, then the test will continue to run without failure. If the first change to HalfTurbo fails, then the test hangs, and nothing else can be done.

Implication: TBD

Workaround: Do not use HalfTurbo Mode. If you wish to run the core at a HalfTurbo Frequency, then perform a full frequency change to the desired frequency. For example, if you wish to run the core at 156 MHz, then set L=8, N=1.5, T=1, HT=0, A=1, B=1.

Status: No Fix

E39. MEMC: Memory Controller hangs when entering Self Refresh Mode.

Problem: If software manually puts SDRAM into Self Refresh Mode, then the memory controller will not perform any more activity.

Here are the steps to recreate the issue:

```
1) MDREFR[K1RUN] = 1                  // make sure SDCLK1 is running.
2) MDREFR[K1FREE] = 0                 // turn off free running.
3) MDREFR[SLFRSH] = 1                 // enter self refresh mode.
4) wait until enter Self Refresh mode.// delay approximately 2 usec.
5) Any non-SDRAM activity will not be performed by the memory controller
```

Implication: Will not be able to access external non-SDRAM memories after the SDRAMs are put into Self Refresh Mode.

Workaround: The user is allowed to assert the SLFRSH bit to the memory controller placing the SDRAMs in Self Refresh Mode, if and only if, the very next transaction the memory controller receives is the deassertion of the SLFRSH bit. Nothing else is permitted on the memory controller after the assertion of the SLFRSH bit, and before the deassertion of the SLFRSH bit. Essentially, this means that the user may perform internal transfers that do not effect the memory controller, such as reads from Instruction Cache, reads/writes to Data Cache, reads/writes to SRAM, reads/writes to internal peripherals, but cannot perform external accesses such as VLIO or PCMCIA

Status: No Fix

E40. SDIO: SDIO Devices Not Working at 19.5 Mbps

Problem: SD/SDIO controller can only support up to 9.75 Mbps data transfer rate for SDIO card. However, the SD/MMC card works fine at 19.5 Mbps clock rate.

The error reason is "HandleEndCommandInterrupt: response for command 52, contains a CRC error". Other commands may respond with a CRC error also.

Many cards were tested and some fail and some pass at 19.5 Mbps, however all cards pass at 9.75 Mbps.

Test was conducted under BSP 3.00.029, V2 Beta that supports full SDIO functionality. Test was conducted in both 1-bit and 4-bit mode and both modes fail. Test was conducted on a Mainstone board.

The Mainstone board has an analog switch in the SDIO path. This switch, according to the spec, can cause up to 18ns of delay. The delay causes the clock to be later at the card and then the data gets delayed going back to the processor. According to the SD spec, the output delay maximum is 15ns @ 25MHz, or 5ns of setup time before the rising edge of the clock. This analog switch was taken out of the circuit, however, cards were still failing.

Implication: TBD

Workaround: Slow the speed of the interface down until the card passes. This speed could be as low as 9.75 Mbps.

Status: Plan Fix

E41. AC97: Command Done bit is never set when data is written in slot12

Problem: The “Codec Done Bit” does not get set upon completion of a write to register 0x54 in Modem IO Space. Moreover, no AC97 controller interrupt will occur to the CPU, telling the software that a register write to 0x54 is complete.

Implication: TBD

Workaround: Software must poll the CAIP bit to determine if a write was accomplished.

Status: No Fix

E42. SD/MMC: SD/MMC controller CRC errors with some SD/MMC cards

Problem: In MMC/SD/SDIO cards, not in SPI mode, CMD2, CMD9, and CMD10 use an R2 type Response. An R2 Response is a 136 bit Response, where:

- [135:128] is 0x3F
- [127:1] is the register being read, where bits [7:1] is the CRC for bits [127:8] of the register
- [0] is the END bit, which is always 1.

The controller ignores bits [135:128] for CRC checking.

The controller should use bits [127:1] for CRC checking. The problem is that the controller is using bits [126:1] for CRC checking. If bit 127 is a 0, then there is no problem, because the CRC checker is always initialized to 0. However, if bit 127 is a 1, then a response CRC error will occur.

Implication: TBD

Workaround: If a Response CRC error occurs for CMD2, CMD9, or CMD10, and bit 127 is a 1, ignore the error.

Status: Plan Fix

E43. USBH: There is no Individual Power Sense Polarity bit for each Host Port. The Power Sense Polarity bit controls the polarity for all three Host Ports.

Problem: Overcurrent indicator signals for Host Port 1 and 2 are controlled externally by USBHPWR[2:1]. Overcurrent indicator signal for Host Port 3 is not pinned out to an external pin. This signal is internally tied to ground.

The Power Sense Polarity bit (UHCHR[PSPL]) controls the polarity of the Overcurrent indicator signals for all three Host Ports.

If a usage model requires that the polarity of the Host Port 1 or 2 be active low, then this will cause a false overcurrent indication to occur on Host Port 3.

Also, software cannot ignore overcurrent on just Host Port 3. Ignoring overcurrent can only be done across all three ports by setting UHCRHDA[NOCP].

The intended silicon fix is to disconnect the Host Port 3 Overcurrent indicator signal from internal ground and connect it to the Power Sense Polarity (PSPL) bit. This guarantees that Host Port 3 will never see an overcurrent condition.

Note that when switching PSPL, a small glitch might be seen on the Overcurrent indicator for Host Port 3. This should not be an issue since the usage model for PSPL is that it is designed to be set statically at initialization time.

Implication: TBD

Workaround: TBD.

Status: Plan Fix

E44. KBD: Keypress wakeup from Standby mode is not reliable

Problem: When the processor is put into Standby mode by correctly configuring the PKWR register, a valid keypress does not reliably bring the processor out of Standby mode.

The failure rate is less than 10 times out of 100 attempts to wakeup from Standby mode via a key press.

Implication: TBD

Workaround: Do not put the processor into Standby mode during keypress activity.

Status: No Fix

E45. USBH: USB Host Port 3 in Transceiverless Mode may not work correctly with an external device.

Problem: The USB Host Port 3, when put into Transceiverless Mode by setting UP3OCR[CFG] = 2, expects a certain behavior from the attached external device.

While the device is in receive mode, i.e. while USB_P3_2 (OE_n) is deasserted, the Host Port 3 expects the device to transmit a steady state “1” on the USB_P3_6 (VPO) pin and a steady state “0” on the USB_P3_4 (VMO) pin.

If the device does not comply to this restriction, then please use a 2-input OR gate, with one input connected to OE_n, the other input connected to VPO from the device, and the output connected to USB_P3_6 (VPO) of the processor.

The intended silicon fix is as follows:

Remove the "Host Port 2 D+ Pull Up Bypass Enable" section of the Developer’s Manual. The D+ Pull Up Bypass Enable is permanently turned off. In other words, the dynamic D+ pullup resistor is always enabled as specified by USBOTG.

This bit position, UP2OCR[6], is now UP2OCR[VPMBlockEnbN].

When VPMBlockEnbN = 0b0 (new functionality):

- When the USB is in transceiverless mode (UP3OCR[CFG] = 0x2), then USB_P3_6 (VPO) and USB_P3_4 (VMO) are ignored by USB Host Port 3 when USB_P3_2 (OE_n) is deasserted.
- However, this will cause the following restrictions:
  - — USB Host Port 3 in Transceiverless Mode can only be connected to a fast speed (12Mbps) device.
  - — The USB device cannot use the common method of indicating a connect by pulling the VPO line high, or disconnect by pulling both VPO and VMO low. The USB device must use another method of indicating a connect/disconnect to the Host Port 3, such as using another GPIO, or performing a read to the external device’s status register.

When VPMBlockEnbN = 0b1 (original functionality):

- When the USB is in transceiverless mode (UP3OCR[CFG] = 0x2), then USB_P3_6 (VPO) and USB_P3_4 (VMO) are not ignored by USB Host Port 3 when USB_P3_2 (OE_n) is deasserted.
- However, this will allow the following conditions:
  - USB Host Port 3 in Transceiverless Mode can be connected to a fast speed (12Mbps) device or slow speed (1.5Mbps) device.
  - The USB device can use the common method of indicating a fast speed device connection by pulling the VPO line high, or a slow speed device connection by pulling the VMO line high, or disconnect by pulling both VPO and VMO low.

Implication: TBD

Workaround: Use a 2-input OR gate, with one input connected to OE_n from the device, the other input connected to VPO from the device, and the output connected to USB_P3_6 (VPO) of the processor.

Status: Plan Fix

E46. SD/MMC: SPI mode commands fail even on cards that are compatible with SPI spec 1.0

Problem: When the SD/MMC controller is put into SPI mode, all read commands where data is transmitted in the DATA token, will fail, even if the card is SPI spec 1.0 compliant.

For example:

- C0 step silicon:
  - CMD9/CMD10 failed on cards compatible with SPI spec 1.0
  - CMD9/CMD10 failed on cards compatible with SPI spec 1.0.1

Another example:

- CMD52 can do 1 byte reads. The read data is returned in the Response Token and there is no data transfer on DAT line. Therefore, this command will pass.
- CMD53 can read multiple bytes and blocks. The read data is returned on the DAT line. Therefore, this command will fail.

Implication: When the SD/MMC controller is put into SPI mode, all read commands where data is transmitted in the DATA token, will fail.

Workaround: Do not use SD/MMC in SPI mode or use Read Commands that return data in the Response Token.

Status: Plan Fix

E47. CLOCKS AND POWER: PWM Clock Enables do not work as specified

Problem: According to the specification, CKEN[0] controls the PWM0 and PWM2 Clock Enable, and CKEN[1] controls the PWM1 and PWM3 Clock Enable. However, CKEN[0] and CKEN[1] both have to be disabled in order to disable any of the PWMx clocks.

CKEN[1:0]	Current Spec	New Spec
0b00	All PWMs disabled	All PWMs disabled
0b01	PWM0, PWM2 enabled	All PWMs enabled
0b10	PWM1, PWM3 enabled	All PWMs enabled
0b11	All PWMs enabled	All PWMs enabled

Implication: TBD

Workaround: To disable a PWMx clock, CKEN[1:0] must be 0b00.

Status: No Fix

E48. ICP: Occasionally EIF, EOF and CRC interrupt are missed when a CRC error is received.

Problem: On some of the product points, occasionally the EIF, EOF, and CRC interrupts are missed when a CRC error is received.

If you read the data out of the RX FIFO, the last piece of data does set the EOF and CRC status bits, but the EIF bit never gets set.

If the interrupt latency is increased by approximately 20usec, the correct behavior is seen. Interrupt latency equals the time from when the ICP generates the interrupt to the time the processor is able to service the interrupt. Latency can be increased by looping on the OS timer until 20us has past.

Implication: If the ICP received a CRC error and is configured to use interrupts, occasionally the EIF, EOF, and CRC interrupts are missed when a CRC error is received.

Workaround: Configure the ICP to use DMA-mode for data transfers. Do not attempt to use interrupt-mode for data transfers.

Status: No Fix

E49. POWER MANAGER: Batt Fault does not always re-enable GPIO 0 and GPIO 1 as wake-up sources.

Problem:

If the 13MHz oscillator is disabled during sleep mode, the processor does not comply to the following statement in section 3.8.1.4 of the *Intel® PXA27x Processor Family Developer's Manual*:

"When nVDD_FAULT or nBATT_FAULT is asserted, PWER assumes its reset value, enabling only GPIO<1:0> as wake-up sources."

If the following sequence is performed:

1. The PWER[1:0] bits are cleared, which disables wake-up due to GPIO<1:0> edge detect
2. The PCFR[OPDE] bit is set, which disables the 13MHz oscillator during sleep mode
3. The processor is put into sleep mode
4. A nVDD_FAULT or nBATT_FAULT is asserted

5. A GPIO<1:0> edge is asserted

then there is no guarantee that the processor will wake from sleep mode.

Implication: There is no guarantee that the processor will wake up from sleep mode, if the PWER[1:0] is cleared, and a nVDD_FAULT or nBATT_FAULT is asserted during the sleep mode, and then a GPIO<1:0> edge is asserted.

Workaround: Any of the following workarounds can be used:

1. Keep PWER bit corresponding to the wakeup enabled before entering sleep.
2. Assert GPIO wakeup signal(s) while the fault is asserted.
3. Keep the 13 MHz oscillator running during sleep mode.

Status: No Fix

E50. POWER MANAGER: The processor does not exit from sleep/deep-sleep mode.

Problem: The processor is exhibiting a problem when waking up from sleep and deep-sleep mode. Specifically, the issue is a race condition between clocks in the processor's sleep/deep-sleep wakeup circuitry. When the processor begins to wake from sleep/deep-sleep mode, a reset to the clock unit de-asserts while the internal power and ground supplies to the wakeup circuitry are not yet stable. The effect is that parts of the clock unit will be in an unknown state after the reset has been de-asserted. In most cases, the race condition will end successfully without de-asserting the reset too early and the issue will not be seen. However, in a small number of parts, the reset de-asserts too early, thus causing the issue.

Implication: Early release of the reset prior to a good, stable power and ground supply can lead to unpredictable values during wakeup and can result in the processor stopping execution. The Erratum is also manufacturing process dependent and intermittent, so only a small percentage of devices are ultimately affected.

Workaround: Intel has found that keeping VCC_Core powered during sleep mode seems to prevent the predictable behavior because it helps keep the correct state of the wakeup circuitry, even when the reset to the wakeup circuitry is de-asserted early. Additional testing has also shown that entering sleep mode at 91 MHz with VCC_Core at 0.95V (+/- 5%), but then grounding VCC_Core during sleep helps to alleviate the issue in a large number of cases, but not all. Finally, the processor's watchdog timer can be implemented to reset the processor, if the failure occurs. This gives customers the ability to gracefully handle a system failure due to this erratum. Use of the potential workarounds can help greatly reduce or possibly eliminate the occurrence of the erratum. Details of these workarounds can be found in the following Application Note: *Sleep/Deep-Sleep Exit Procedure for the Intel® PXA27x Processor Family.*

Status: Plan Fix

E51. SDIO: CMD53 multiple-block data transfer with block count set to 0 not supported.

Problem: A block count of 0 for CMD53 is not supported.

Implication: The block count field in CMD53 is 9 bits (511 maximum blocks). To send more than 511 blocks, set the block count in CMD53 to 0 in the command argument. The PXA27x SDIO controller does not support this functionality.

Workaround: If more than 511 blocks are to be transferred, software should send multiple CMD53 multi-block transfers with a block count that is <= 511.

Status: No Fix

E52. LCDC: Disable-done interrupt does not always occur.

Problem: The LCD controller disable-done and quick-disable-done interrupts are not always issued the second time the LCD controller is enabled and then disabled.

Implication: During normal LCD controller operation, setting the LCD-disable LCCR0[DIS] bit = 1 or LCD-controller-enable LCCR0[ENB] bit = 0 can be used to disable the LCD controller. After the LCD controller has been disabled, a disable-done or quick-disable-done interrupt is issued if their respective mask bits are not enabled. After the LCD controller has been enabled and then disabled for the second time, the disable-done or quick-disable-done interrupts are not issued.

Workaround: Poll the LCD-disable-done flag, LCSR0[LDD], instead of relying on the disable-done or quick-disable-done interrupts.

Status: No Fix

E53. UDC: RCV can not be tied high during UDC transmission when using an external transceiver.

Problem: When the RCV signal is held high during an IN transmission, the UDC may stop transmitting future packets.

Implication: RCV is one of the signals used for the single-ended USB interface option. The RCV input signal is the resultant signal that represents the conversion of the differential pair D+ and D- to a single digital signal. When a USB host makes a request for an IN transmission, if RCV is held high during the IN transmission, the UDC may not acknowledge subsequent ACKs from the host.

Workaround: During an IN transmission, insure that RCV is either held low, or sees the reflection of VP transmit data. VP is another one of the digital signals that is part of a single-ended USB interface. VP is the digital version of the differential signal D+.

Status: No Fix

E54. AC97: AC97 CAR[CAIP] bit field can be incorrectly set.

Problem: 1) If software reads the CAR register in the same cycle that the AC97 unit internally clears the lock, the lock is re-acquired in the same cycle. The (new) locked status is returned to software as the value of CAR. Software never detects that the lock was released.

The AC97 controller provides a bit, CAR[CAIP], that can be used by software as a flag to indicate the AC-link interface is currently busy with an I/O cycle. If you read this bit and get a 0, you have the “lock” and it is safe to use the AC Link. The act of reading the CAR[CAIP] bit sets it to 1. When software reads the CAR[CAIP] bit and sees a 1, that means the AC-link interface is already busy with another driver’s I/O cycle.

2) When software reads an AC97 CODEC register other than register 0x54, two AC-link reads occur. The CAR[CAIP] bit gets cleared (by hardware) after each AC-link read. Since the second AC link read is still in progress, the CAR[CAIP] should not be cleared at this point.

Implication: 1) This results in software looping on a wait-for-CAR[CAIP]-free loop.

2) N/A

Workaround: 1) S/W should not read the CAR[CAIP] bit to determine if a CODEC transaction is in progress

2) S/W should not read the CAR[CAIP] bit to determine if a CODEC transaction is in progress

See Table 6, “Recommended Procedure for Accessing AC97 CODEC Registers” .

Table 7. Recommended Procedure for Accessing AC97 CODEC Registers

Operation	Step by step	IO Completion Detection	End of IO clean up
Read a codec register other than register 0x54	1. Read memory space - discard this data 2. Wait for the IO to complete 3. Read memory space again - save this data 4. Wait for the IO to complete 5. Return the saved data	Wait for GSR:SDONE to assert for step 2 Wait for GSR:SDONE to assert for step 4	Clear GSR:SDONE and GSR:CDONE after step 4
Write a codec register other than register 0x54	1. Write to memory. 2. Wait for IO to complete.	Wait for GSR:CDONE to assert for step 2	Clear GSR:CDONE after step 2
Read Modem register 0x54	1. Read from memory for register 0x54. 2. Wait for the IO to complete.	Wait for GSR:SDONE to assert for step 2	Clear GSR:SDONE and GSR:CDONE after step 2
Write Modem register 0x54	1. Write to memory for register 0x54. 2. Wait for the IO to complete	Wait 50 microseconds for step 2.	None

Note: Acquiring the CAR[CAIP] lock is no longer recommended

Status: No Fix

Specification Changes

Note: There are no specification changes at this time.

Specification Clarifications

Note: There are no specification clarifications at this time.

Documentation Change

Note: There are no documentation changes at this time.